

REMARKS

Claims 1-7 and 21-28 are pending.

In the present Office Action, claims 1-7 and 21-28 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over newly cited reference U.S. 6,687,760 (hereinafter “Bracha”), in view of U.S. Patent No. 6,128,717 (hereinafter “Harrison”). Applicant has reviewed the cited art and believes the pending claims recite a combination of features neither disclosed nor suggested by the art. Accordingly, Applicant traverses the above rejections and requests reconsideration in view of the following comments.

It is first noted that the presently claimed invention is directed to an entirely different system than that disclosed by the cited art. Consequently, the following comments regarding the nature and context of the invention may serve to clarify the nature of the claimed invention.

Generally speaking, the claimed invention is directed to storage devices in storage systems, and storage objects stored therein. For example, note the following selected excerpts from the description:

**“Storage networking** is the practice of **connecting storage devices** to computing devices (e.g., clients, servers, and the like) by using Fibre Channel networks instead of traditional point-to-point small computer system interface (SCSI) channels. A network used to connect servers to storage devices is referred to as a storage area network (SAN). Within a **SAN environment**, all computing devices have **access to the available storage devices**. . . .

Prior to the development of SAN technology, local and wide area networks provided connectivity between computing devices that did not include storage devices. Connections were established with network protocols . . . Distributed File Systems such as network file system (NFS) and Common Internet file system (CIFS) are layered on top of network protocols. Distributed File Systems organize **access to files and correspondingly data storage elements** across a network consisting of heterogeneous computing devices. The computing

devices are typically organized as clients and servers, in a client-server architecture. **Access to files or data storage elements** is transparent to any particular computing device, such that access is consistent across the distributed file system without the need to have any private information about the physical locations or details associated with any particular file or data element.

. . . the very purpose of file system and volume manager function within an operating system is to **hide the private information related to data storage elements located on one or more data storage devices**. Accordingly, operating system vendors, file system vendors or volume manager vendors do not reveal or provide any useful interfaces that provide access to private information. Moreover, storage environment software stacks are complex and trying to extract the private information from existing stacks is not readily achievable without intervention from the file system software or volume manager software itself.

Moreover private information about data storage elements is in a continual state of flux in modern data storage architectures, since at any moment in time data storage elements are moved during storage device failure, device reorganization to reduce fragmentation, and the like. . . .

In an effort to address some of these problems some industry associations have been initiated in an attempt to **standardize data storage device communication**. For example, the Storage Network Industry Association (SNIA) and the National Committee for Information Technology Standards (NCITS) technical committee T11 have been established. . . .

Additionally, even with persistent and stable representations of data storage elements, workable and useable application programming interfaces (APIs) will need to be established, such that different levels of abstraction and interfacing to the storage elements can be achieved seamlessly with user-defined software applications. In this way, user-defined software applications can utilize the APIs to better interact with the storage objects.

. . .

Moreover, user-defined applications are implemented at different levels of abstraction and thus a user-defined application can make reference to a storage element at different abstraction levels within the storage environment. As a result, any provided API must be capable of determining the storage element's reference level within the storage environment. Therefore, access to the private information or absolute

location of the storage element presents a number of challenges when providing APIs to user-defined software applications.

Therefore, what is needed are methods and systems for providing access to **data storage elements residing in any storage environment**, regardless of any reference level within which access is attempted on the storage element, thereby resulting in access to data storage elements becoming more seamless, transparent, flexible, and reliable across storage environments.” (Description, pages 2-6). (emphasis added).

In addition to the above, the description includes:

“Furthermore, as used herein a computing device includes one or more processing elements coupled with computer readable memory which can be volatile or nonvolatile memory or any combination thereof. Additionally, the term **"object" or "storage object" as used herein includes data storage elements such as, and by way of example only electronic files, portions of data related to a single electronic file, a file system, a database, a storage device partition, and the like.**” (Description, page 9, lines 7-12).

Other similar descriptions occur throughout the specification. In view of the above, and the remainder of the description, it is believed the context of the claimed invention is very clearly that of storage devices and objects stored therein.

In contrast to the context of the presently claimed invention, Bracha is directed to object oriented programming, and in particular “to a method and apparatus for performing method lookup to support transitive method override in the presence of modularity constructs.” (Bracha, col. 1, lines 10-12). Applicant believes one skilled in the art would immediately recognize the distinct nature of the presently claimed invention and Bracha.

Currently pending claim 1 recites:

“A method for resolving a **storage object's** absolute location **within a first storage environment** to grant access to the storage object, comprising: receiving a **storage object reference**;

determining an initial storage management stack level associated with the storage reference;  
iterating through one or more additional storage management stack levels beginning with the initial stack level in response to determining the storage reference is not an absolute reference; and  
translating the storage reference through each iteration into one or more relative extents until one or more absolute extents are obtained, wherein the one or more absolute extents comprise the storage object's absolute location within the first storage environment.” (emphasis added).

In paragraph 2 of the present Office Action, it is stated:

“Bracha discloses a method for resolving a storage object’s absolute location within a first storage environment to grant access to the storage object, comprising: receiving a storage object reference (abstract; summary; col. 5, lines 39-55 and line 57 to col. 6, line 23, Bracha) . . .”

However, Applicant submits Bracha does not disclose the above recited features. It would appear that the examiner is equating Bracha’s object oriented “object” with the recited “storage object”. However, in view of the above discussion, Applicant submits they are clearly not equivalent. In Bracha, and in the object oriented sense, it is well known by those skilled in the art that an “object” is, generally speaking, a memory resident instance of a data structure and operations defined by the object's class. However, Applicant submits it is equally clear to one skilled in the art that the recited “storage object” is not an object in the object oriented sense. While a claim term may be presumed to have its ordinary and customary meanings, the ordinary and customary meaning is by one of ordinary skill in the art. Applicant believes the term “storage object”, which is also provided definition in the description as noted above (i.e., a storage object is a file, portion of a file, file system, database, storage device partition, and the like), is clearly understood by one skilled in the art to be entirely different from an object oriented “object.” Accordingly, Applicant submits Bracha does not disclose “receiving a storage object reference” as recited and does not disclose the features as suggested in paragraph 2 of the present Office Action.

In addition to the above, Bracha does not disclose “determining an initial storage management stack level associated with the storage reference.” Rather, Bracha discloses a “path stack” which lists a class hierarchy between a target object and a resolved class. Applicant submits the “path stack” of Bracha is not equivalent to the recited “storage management stack”. Rather than being a storage management stack as recited, the path stack of Bracha is used to “determine which implementation of [a] method m to invoke in response to the method invocation.” (Bracha, col. 3, lines 5-7). Accordingly, Applicant believes the feature “determining an initial storage management stack level associated with the storage reference” is patentably distinct from the cited art.

In addition to the above, Applicant submits that the combination of Bracha and Harrison does not teach all of the features of the presently claimed invention. As already discussed, Bracha discloses a method and apparatus for performing object oriented programming method lookup in order to support method override in the presence of modularity constructs. As noted, objects in Bracha are objects in the object oriented programming sense. Harrison discloses a hard disk based application programming interface which is directed to storing data on a disk based on the type or size of the data and characteristics of the available storage on the disk. For example, Harrison discloses:

“Another more specific object of the present invention is to provide a SAPI implementation engine within a mass storage device such as a disk drive which determines and attaches a particular SAPI descriptor to a data object and which periodically revises a global storage strategy for the device based upon SAPI descriptors and record extents presently in storage and physically rearranges the records to achieve improved storage performance at the host level.” (Harrison, col. 6, lines 1-8).

Accordingly, while both Bracha and Harrison may use the word “object”, Bracha and Harrison are directed to fundamentally different concepts and bear no relation to one another. As already discussed, Bracha discloses a method for use in an object oriented programming environment wherein a “path stack” is used to identify a class hierarchy between a target object and a resolved class. Harrison concerns a storage device API for

determining a more efficient approach to storing data on a disk. For example, Harrison describes various aspects of mapping data on a drive such as:

“Four disks 30A, 30B, 30C and 30D are shown in stacked relationship upon a rotating spindle in the FIG. 2 hard disk drive 10. Each disk 30 defines a multiplicity of concentric data tracks. As shown in FIG. 3 the tracks are arranged into e.g five radial bands or zones A, B, C, D, and E, with each zone having a data transfer rate adjusted for relative head-to-disk velocity at an inside radius of the particular zone.” (Harrison, col. 7, lines 40-47).

“Alternatively, and perhaps less satisfactorily, a disk drive storage command 6 which is delivered to the drive 10 separate from the data object 5 may have a format as shown in FIG. 6. This command format 6 would typically include a device identifier field 56, an operation or command field 58 (such as read/write, etc.) a starting LOA field 60 and a record unit or block extent field 62.” (Harrison, col. 10, lines 5-11).

As can be seen, Harrison and Bracha are completely unrelated, and one would not be motivated to look to the hard drive API of Harrison to modify the object oriented programming method disclosed in Bracha. Further, the combination of Bracha and Harrison clearly does not disclose the features of the presently claimed invention. Rather, such a hypothetical combination would merely provide an object oriented method for identifying a class hierarchy between a target object (i.e., instance of a class) and a resolved class, and the hard disk API of Harrison. However, there is no connection between the two. Rather, they are completely independent mechanisms.

Applicant submits the claims are patentably distinct from the cited art and the application is in condition for allowance. However, should the examiner believe issue remain that would prevent the present application from proceeding to allowance, the below signed representative requests a telephone call at (512) 853-8866 in order to facilitate a speedy resolution.

**CONCLUSION**

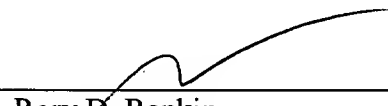
Applicant submits the application is in condition for allowance, and an early notice to that effect is requested.

If any extensions of time (under 37 C.F.R. § 1.136) are necessary to prevent the above referenced application(s) from becoming abandoned, Applicant(s) hereby petition for such extensions. If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5760-17100/RDR.

Also enclosed herewith are the following items:

☒ Return Receipt Postcard

Respectfully submitted,



---

Rory D. Rankin  
Reg. No. 47,884  
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin,  
Kowert, & Goetzel, P.C.  
P.O. Box 398  
Austin, TX 78767-0398  
Phone: (512) 853-8800

Date: July 29, 2005